2008-04-01

Defense Acquisition Management for Systems-of-s

Shayani Ghose

NPS-AM-08-033

EXCERPT FROM THE

PROCEEDINGS

OF THE

FIFTH ANNUAL ACQUISITION RESEARCH SYMPOSIUM

DEFENSE ACQUISITION MANAGEMENT FOR SYSTEMS-OF-SYSTEMS

Published: 23 April 2008

by

Dr. Daniel DeLaurentis and Shayani Ghose

5th Annual Acquisition Research Symposium of the Naval Postgraduate School:

Acquisition Research: Creating Synergy for Informed Change

May 14-15, 2008

Prepared for: Naval Postgraduate School, Monterey, California 93943

ACQUISITION RESEARCH PROGRAM
GRADUATE SCHOOL OF BUSINESS & PUBLIC POLICY
NAVAL POSTGRADUATE SCHOOL

The research presented at the symposium was supported by the Acquisition Chair of the Graduate School of Business & Public Policy at the Naval Postgraduate School.

To request Defense Acquisition Research or to become a research sponsor, please contact:

NPS Acquisition Research Program
Attn: James B. Greene, RADM, USN, (Ret)
Acquisition Chair
Graduate School of Business and Public Policy
Naval Postgraduate School
555 Dyer Road, Room 332
Monterey, CA 93943-5103
Tel: (831) 656-2092
Fax: (831) 656-2253
E-mail: jbgreene@nps.edu

Copies of the Acquisition Sponsored Research Reports may be printed from our website www.acquisitionresearch.org

Conference Website:
www.researchsymposium.org

ACQUISITION RESEARCH PROGRAM
GRADUATE SCHOOL OF BUSINESS & PUBLIC POLICY
NAVAL POSTGRADUATE SCHOOL

Proceedings of the Annual Acquisition Research Program

The following article is taken as an excerpt from the proceedings of the annual Acquisition Research Program. This annual event showcases the research projects funded through the Acquisition Research Program at the Graduate School of Business and Public Policy at the Naval Postgraduate School. Featuring keynote speakers, plenary panels, multiple panel sessions, a student research poster show and social events, the Annual Acquisition Research Symposium offers a candid environment where high-ranking Department of Defense (DoD) officials, industry officials, accomplished faculty and military students are encouraged to collaborate on finding applicable solutions to the challenges facing acquisition policies and processes within the DoD today. By jointly and publicly questioning the norms of industry and academia, the resulting research benefits from myriad perspectives and collaborations which can identify better solutions and practices in acquisition, contract, financial, logistics and program management.

For further information regarding the Acquisition Research Program, electronic copies of additional research, or to learn more about becoming a sponsor, please visit our program website at:

www.acquistionresearch.org

For further information on or to register for the next Acquisition Research Symposium during the third week of May, please visit our conference website at:

www.researchsymposium.org

THIS PAGE INTENTIONALLY LEFT BLANK

Defense Acquisition Management for Systems-of-systems

Presenter: Shayani Ghose is a graduate student in the School of Aeronautics and Astronautics Engineering, Purdue University. She received her BS in Electrical Engineering and Computer Engineering (Dual) from Drexel University, Philadelphia, in June 2007. She is currently a part of the System-of-systems Research Group advised by Dr. Daniel DeLaurentis.

Shayani Ghose
Graduate Research Assistant, School of Aeronautics and Astronautics
Purdue University
701 W. Stadium Avenue
West Lafayette, IN 47907
Ph : +1 765 494 7958
Fax: +1 765 494 0307
Email: ghose@purdue.edu

Presenter: Daniel DeLaurentis is an Assistant Professor in the School of Aeronautics and Astronautics Engineering, Purdue University. He received his PhD from Georgia Institute of Technology in Aerospace Engineering in 1998. His current research interests are in Mathematical modeling and object-oriented frameworks for the design of system-of systems, especially those for which air vehicles are a main element and approaches for robust design, including robust control analogies and uncertainty modeling/management in multidisciplinary design.

Daniel DeLaurentis
Assistant Professor, School of Aeronautics and Astronautics
Purdue University
701 W. Stadium Avenue
West Lafayette, IN 47907
Ph : +1 765 494 0694
Fax: +1 765 494 0307
Email: ddelaure@purdue.edu

Abstract

The Department of Defense (DoD) has placed a growing emphasis in recent years on the pursuit of agile capabilities via net-centric operations. Dramatic technological advancements in communications and sensing have generated opportunities for battlefield systems to exploit collaboration for multiple effects. In this setting, systems are expected and often required to interoperate along several dimensions. Yet, the manner in which these "system-of-systems" are acquired (designed, developed, tested and fielded) has not kept pace with the shifts in operational doctrine. Systems acquisition remains largely focused on requirements for individual operation, paying insufficient attention to the ability of systems to influence the variety of future ecosystems in which they may subsist. Further, acquisition programs have struggled with complexities in both program management and engineering design. This paper establishes an understanding and classification of underlying complexities in the acquisition of system-of-systems. It also provides a conceptual model that exposes the connectivity between systems and the impact of system heterogeneity and externalities on that connectivity throughout the acquisition lifecycle. Implementation of this model in an exploratory simulation is in progress. Its objective is to allow acquisition professionals to develop intuition for procuring and deploying system-of-systems, providing a venue for experimentation and exploration to develop insights that underpin successful acquisition of SoS-oriented defense capabilities.

1. Introduction

A system-of-systems (SoS) consists of multiple, heterogeneous, distributed systems that can (and do) operate independently but can also assemble in networks and collaborate to achieve a goal. According to Maier (1998), component systems of the SoS typically demonstrate traits of operational and managerial independence, emergent behavior, evolutionary development and geographic distribution. Networks of component systems often form among a hierarchy of levels and evolve over time as systems are added to or removed from the SoS. However, these component systems are often developed out of context of their interactions with the future SoS. As a result, the systems may be unable to interact with the future SoS, adapt to any emergent behavior, or be robust in the face of external disturbances. The US Coast Guard's (USCG) Integrated Deepwater System (IDS) is an example of a Department of Homeland Security (DHS) acquisition process for an SoS, "patterned after the successful Department of Defense (DoD) model of contracting to competing industry teams" (Anderson, Burton, Palmquist, & Watson. 1999). IDS has faced technical and management challenges similar to those that are historically prevalent in acquisitions in SoS environments.

In the 1990s, the USCG Acquisition Directorate recognized the need to "deliver and support new generations of platform and mission systems" (1998). The 25-year, $24 billion IDS is aimed at "delivering new aircraft and cutters, modernizing legacy assets, and providing a new generation of Command, Control, Communications, Computers, Intelligence, Surveillance and Reconnaissance (C4ISR) mission systems for forces deployed and ashore" (1998). In 2002, the Coast Guard awarded this contract to Integrated Coast Guard Systems (ICGS), an industry consortium of Northrop Grumman, Lockheed Martin and several other defense contractors. ICGS was contracted to act as the Lead System Integrator responsible for acquiring assets and integrating them into the IDS. The USCG recently dismissed its Lead System Integrator after a series of technical and managerial failures (Allen, 2007). In 2006, the Government Accountability Office (GAO) reported that the collaboration among the subcontractors continues to be problematic and that the system integrator wields little influence to compel decisions among them (Caldwell, 2006).

The DHS and the DoD are not the only organizations struggling with systems integration of a collection of complex system. The Air Transportation System and the NASA Constellation program are also facing similar challenges in attempting to apply generic system engineering processes for acquisition in an SoS environment. Integration challenges faced by the Constellation Program are documented in a recent NRC report (Committee on Systems Integration for Project Constellation, 2004). Both DoD and non-DoD examples are the key drivers motivating the research described in this paper.

The overarching goal of this research is to understand which types of acquisition management, policy insights and approaches can increase the success of an acquisition process in the SoS setting. The three research questions being explored are as follows:

1. Is there a taxonomy by which one can *detect* classes of complexities in particular SoS applications?
2. What are the underlying systems engineering (SE) and program management functions that are affected?
3. How can exploratory modeling generate SE and acquisition management insights and approaches to improve the probability of success?

In order to answer some of the questions posed, we aim to

1. Identify the complexities in the acquisition of SoS based on historical trends of "failures," especially in the context of the DoD and DHS.
2. Develop a conceptual model of a generic acquisition process that can then be customized to different SoS applications.
3. Develop and simulate a computational model using an existing acquisition process for an SoS as a case-study; for example, the USCG Integrated Deepwater System could be used as an example of a DHS acquisition process set in an SoS environment. Interpretations of the results obtained would be used to field the research questions posed.

Since the project is presently at its midway point, in this paper we only focus on the first two research questions, specifically, on the mappings between SoS acquisition difficulties and complexities with a view toward model development. A general framework for and an outline of the computational model are provided.

2. Mapping Failure Modes to Underlying Complexities

Simon (1996) and Bar-Yam (2003) define complexity as the amount of information necessary to describe a system effectively. In the context of a system-of-systems, the necessary information encompasses both the systems that comprise the SoS and their time-varying interactions with each other and the "externalities." Rouse suggested that the complexity of a system (or model of a system) is related to

- The intentions with which one addresses the systems.
- The characteristics of the representation that appropriately accounts for the system's boundaries, architecture, interconnections and information flows.
- The multiple representations of a system, all of which are simplifications; hence, complexity is inevitably underestimated.
- The context, multiple stakeholders, and objectives associated with the system's development, deployment and operation. (2007)
- (Polzer, DeLaurentis and Fry (2007) explored the issue of multiplicity of perspectives, in which perspective was defined as a system's version of operational context.)
- The learning and adaptation exhibited during the system's evolution (Rouse, 2007).

Historical data from previous unsuccessful defense acquisition programs show a distinct correlation with the causes for complexity identified by Rouse (2007). Fowler (1994) points out some of the causes for the failure of the Defense Acquisition Process to be "over specification and an overly rigid approach on development": unreasonably detailed cost estimates of development and production, impractical schedules and extremely large bureaucratic overhead. Dr. Pedro Rustan, director of advanced systems and technology at the National Reconnaissance Office, identified four specific shortcomings in the acquisition process for defense space systems: "initial weapons performance requirements that are too detailed and lacking flexibility," "insufficient flexibility in the budget process," "a propensity to increase performance requirements in the middle of the acquisition cycle" and demands to field entirely new spacecraft to meet new requirement" (Spring, 2005). Riccioni (2005) used the United State Air Force (USAF) F-22 Raptor Program to illustrate shortcomings of the existing Defense Acquisition Process. Some of the recognized reasons for the failure of the F-22 Raptor Program were the ambitious nature of the set requirements, the "gross underestimation" and continual

misrepresentation of cost in order to "seduce the Congress and the Public" to believe that the aircraft was affordable. However, the major failing of this program lies in that the enormous delay in the development process (spanning over two decades) resulted in an aircraft that was no longer needed since the "existing and future enemies changed natures." Riccioni also points out that "terrorists are the only extant and foreseeable threats" but they do not threaten the West with fighter aircrafts. In another example, the US Army's Future Combat Systems (FCS) has found difficulties in developing and fielding equipment that meet the program objective (Capaccio, 2006).

Using the above examples in conjunction with other acquisition programs, such as USCG Integrated Deepwater System and Future Combat System, we summarize the common causes of failure within acquisition processes as: a) *misalignment* of objectives among the systems, b) limited *span of control* of the SoS engineer on the component systems of the SoS, c) *evolution* of the SoS, d) *inflexibility* of the component system designs, e) *emergent behavior* revealing hidden dependencies within systems, f) *perceived complexity* of systems and g) the challenges in *system representation*.

Sage and Biemer (2007) examined the existing systems engineering process models in the context of their applicability to SoS and concluded that none of them "could be tailored to systems family development." Sage and Biemer also developed the System-of-systems Engineering (SoSE) Process Model designed specifically for SoS applications. The complexities discussed above were mapped onto a section of the SoSE Process Model based on the trends observed in past acquisition processes within the DoD Acquisition Process. Figure 1 depicts the mapping of some of these complexities to a section of the SoSE Process Model, representing from where complexities might arise and how they may affect the acquisition process. For example, SoS operations could demonstrate emergent behavior and result in a change in the CONOPs for the SoS. Evolution of the SoS changes the CONOPS of the SoS may result in a subsequent change in the Acquisition Strategy. Misalignment of objectives of the component systems in an SoS can arise from both the CONOPs as well as the SoS Project Control. System inflexibility, perceived complexities and challenges in representing systems occur mostly between or within systems. Accurate representation of component systems is complicated by the presence of hidden and visible dependencies between systems, fuzzy boundaries, unknown architectures, etc.

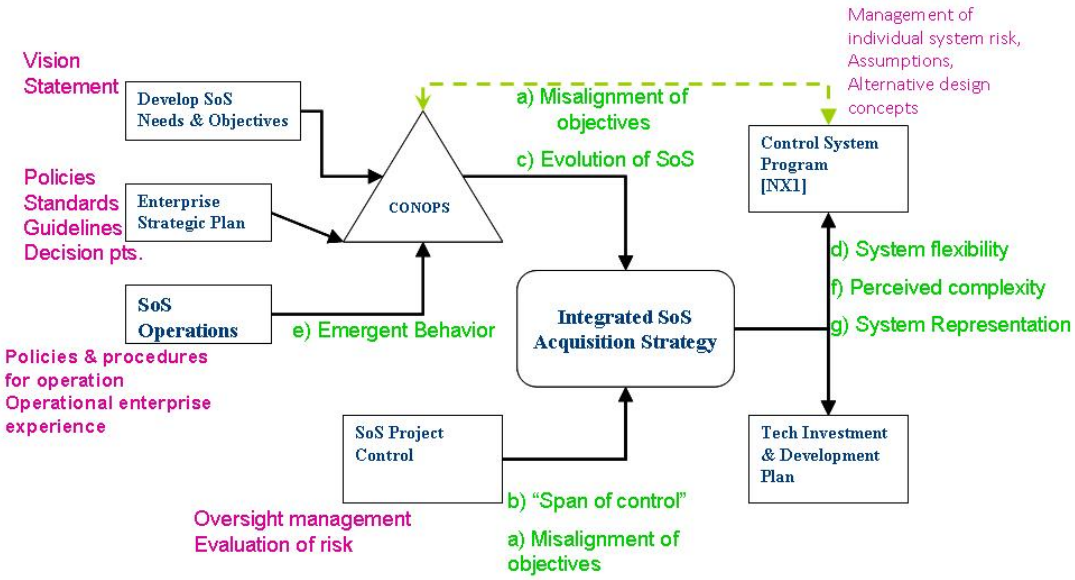

Figure 1. Complexities Mapped to a Section of the SoSE Process Model
(Sage & Biemer, 2007)

3. Towards Development of an Exploratory Model for SoS Acquisition

3.1 Pre-Acquisition Model

The purpose of the pre-acquisition model is to better understand the external stakeholders that affect the acquisition process. The model we developed is depicted in Figure 2 and is based loosely on Sage and Biemer (2007) SoSE Process Model. External inputs to the SoS acquisition process are sorted into three categories: "Capabilities & Possibilities" (CAP), "Technology Assessment, Development, Investment and Affordability Plan" (ADIA) and the funding received. The CAP and ADIA are our own creation. Though they are similar to the *Concepts of Operation* (CONOPs) and *Technology Investment and Development Plan* (TDIP) in the SoSE Process Model, there are some key differences.

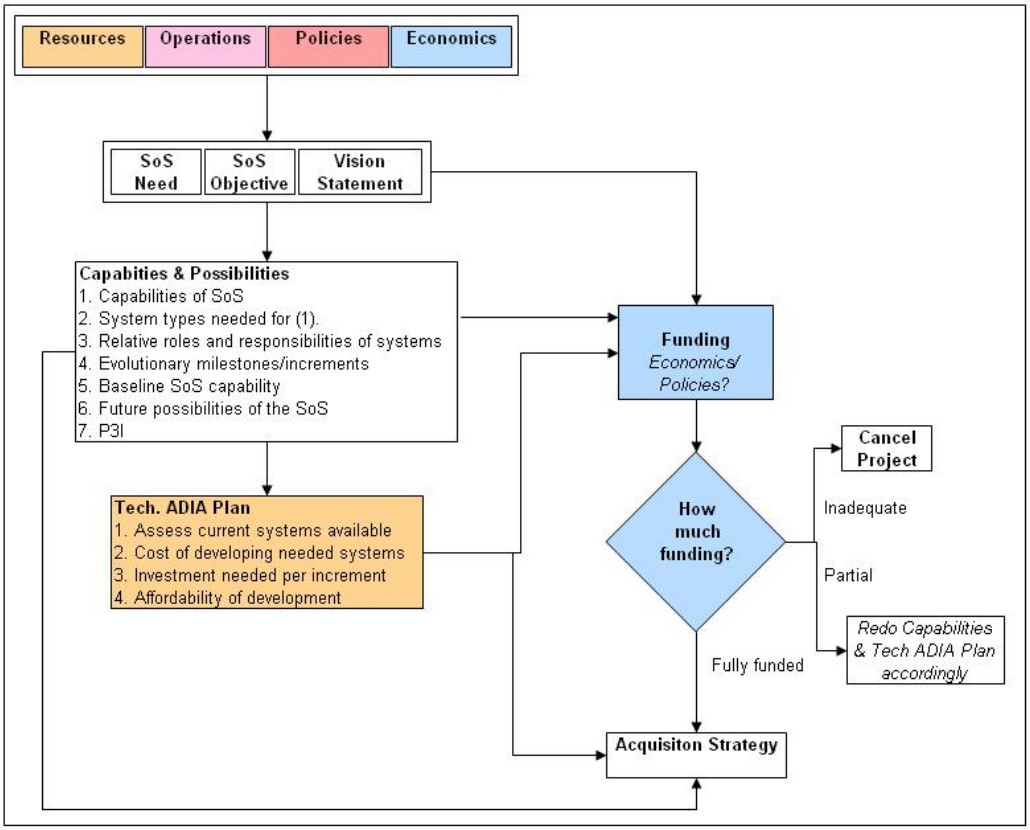

Figure 2. Conceptual Model for Pre-Acquisition Activities

The need, objective, and vision for an SoS feed the CAP. The CAP is a high-level requirements document that provides the following information:

1. The capabilities that the SoS is required to possess and services it must provide
2. The system types that are needed to provide these capabilities
3. The relative roles and responsibilities of the constituent systems
4. Milestones in the development of the SoS and the number of increments needed
5. Baseline SoS capability at its first deployment
6. Future possibilities for the SoS in terms of capabilities it may possess and services it may provide
7. Pre-planned product improvement (P3I) for each system type to support future capabilities of the SoS

The main differences between the CONOPS and CAP stem from the last two entries (6 and 7). The evolutionary nature of the SoS requires the dynamic addition and removal of component systems and functionalities. While the capabilities required for the SoS at the time of first deployment may be basic, the future capabilities of the SoS allow the systems of the SoS to be developed keeping in mind the future capabilities the SoS will provide. This prevents individual systems from becoming obsolete or being drastically re-designed.

The CAP feeds the Technology ADIA Plan, which is needed to

1. Assess the capabilities of the legacy systems and their current maturities
2. Provide a cost estimate for upgrading/ developing systems
3. Provide an estimate of the investment that will be needed per increment
4. Assess the affordability of the investment

The differences in the Technology ADIA Plan and the TDIP stem from entries 2 and 3 in the list above. In addition to the functions of the TDIP, the Technology ADIA Plan provides cost estimates for all new development needed, including a cost breakdown per increment in the development process.

Both the CAP and the Technology ADIA Plan are required to determine the amount of funding required for the project. While there are numerous factors that are used to determine funding (such as political affiliation, unexpected crisis, regulations etc), CAP and the Technology ADIA Plan are the inputs to the acquisition process that translate into technical requirements for the SoS. Provision of a computational model of the pre-acquisition activities is outside the scope of this paper. Instead, we focus on realizing a model for the acquisition strategy, which is described next.

3.2 Acquisition Strategy Model

Development of a "brand new" SoS has been and will remain a rare occurrence. The United States Air Force (USAF) Scientific Advisory Board (Saunders et al., 2005) states that one of the challenges in building an SoS is that legacy systems are contributors that affect the performance of other systems. These legacy systems, may be used "as-is" or may need some re-engineering to feed the needs of the new SoS. In addition, new systems are incorporated to develop the capabilities of the SoS. Again, the new systems can range from off-the-shelf, plug-and-play products to custom-built systems dependent of the working of a legacy system. The breadth of the heterogeneity of the components can be broadly categorized under legacy systems, new systems and improvements. Sub-categories arise when two or more categories overlap.

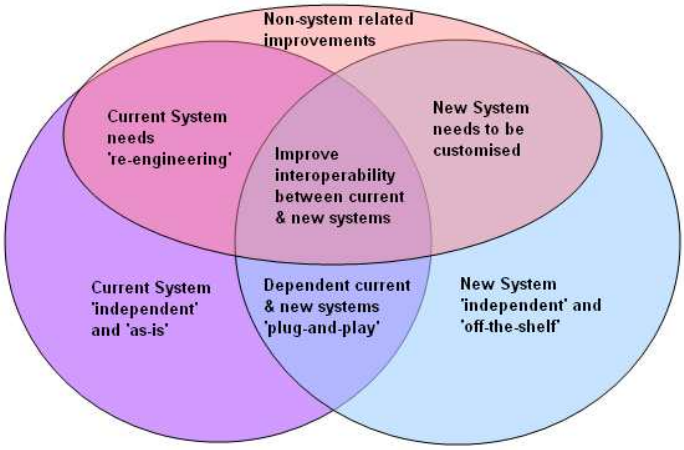

Figure 3. Heterogeneity of Component Systems in an SoS

The different components that comprise the acquisition process in an SoS environment are depicted in Figure 3. For example, improvements can be non-system related, such as improvements in business practices for the SoS, or they can be system-related such as re-engineering legacy systems or customizing/developing new systems to meet the needs of the SoS. Similarly, legacy and new systems can be independent "as-is" systems, dependent "as-is" systems, independent systems in need of "re-engineering," or dependent systems that need customization. Another subcategory is based on the interoperability of the systems in the SoS. While some dependent systems have existing interfaces that allow them to be interoperable (plug-and-play), others need to develop interfaces that allow them to interact with other systems.

Implementing and integrating these different kinds of systems and processes into the SoS is made more complex by the evolutionary nature of an SoS. Thus, it must be made possible for component systems to re-designed or upgraded dynamically without having to re-design the entire SoS. Also, though most systems depend on others during the implementation or integration phases, they are not centrally controlled. This requires that the systems have an incentive to collaborate with each other without being forced to do so. These issues are merely a sub-set of the challenges for an acquisition process (as discussed in Section 2) in an SoS environment.

The conceptual model for acquisition strategy proposed in this section is based on the 16 basic technical management and technical system-engineering processes outlined in the Defense Acquisition Guidebook (DoD, 2006), often referred to as the 5000-series guide. However, an SoS environment changes the way these processes are applied. The 2007 System-of-Systems System Engineering (SoS-SE) Guide (Systems and Software Engineering, 2006) addresses these considerations by modifying (or in some cases revamping) some of the 16 processes in accord with an SoS environment. These new processes and their functions are described in Table 1. Our conceptual model for acquisition in an SoS environment is illustrated in Figure 4. It is centered on the revised processes and depicted in a hierarchy to show the flow of control between the processes throughout the acquisition lifecycle.

Table 1. Modified Technical Management and Technical Processes, as Dscribed in the DoD SoS-SE Guidebook
(Systems and Software Engineering, 2006)

Requirements Development	Takes all inputs from relevant stakeholders and translates the inputs into technical requirements
Logical Analysis	Is the process of obtaining sets of logical solutions to improve the understanding of the defined requirements and the relationships among the requirements (e.g., functional, behavioral, temporal)
Design Solution	Process that translates the outputs of the Requirements Development and Logical Analysis processes into alternative design solutions and selects a final design solution.
Decision Analysis	Provides the basis for evaluating and selecting alternatives when decisions need to be made.
Implementation	The process that actually yields the lowest level system elements in the system hierarchy. The system element is made, bought or reused.
Integration	The process of incorporating the lower-level system elements into a high-level system element in the physical architecture.
Verification	Confirms that the system element meets the design-to or build-to specifications. It

	answers the question, “Did you build it right?”
Validation	Answers the question “Did you build the right thing?”
Transition	The process applied to move the end-item system to the user.
Technical Planning	Ensure that the systems engineering processes are applied properly throughout a system’s lifecycle.
Technical Assessment	Activities measure technical progress and the effectiveness of plans and requirements.
Requirements Management	Provides traceability back to user-defined capabilities
Risk Management	To help ensure program cost, schedule and performance objectives are achieved at every stage in the lifecycle and to communicate to all stakeholders the process for uncovering, determining the scope of, and managing program uncertainties.
Configuration Management	The application of sound business practices to establish and maintain consistency of a product’s attributes with its requirements and product configuration information.
Data Management	Address the handling of information necessary for or associated with product development and sustainment.
Interface Management	Ensures interface definition and compliance among the elements that compose the system, as well as with other systems with which the system or systems elements must interoperate.

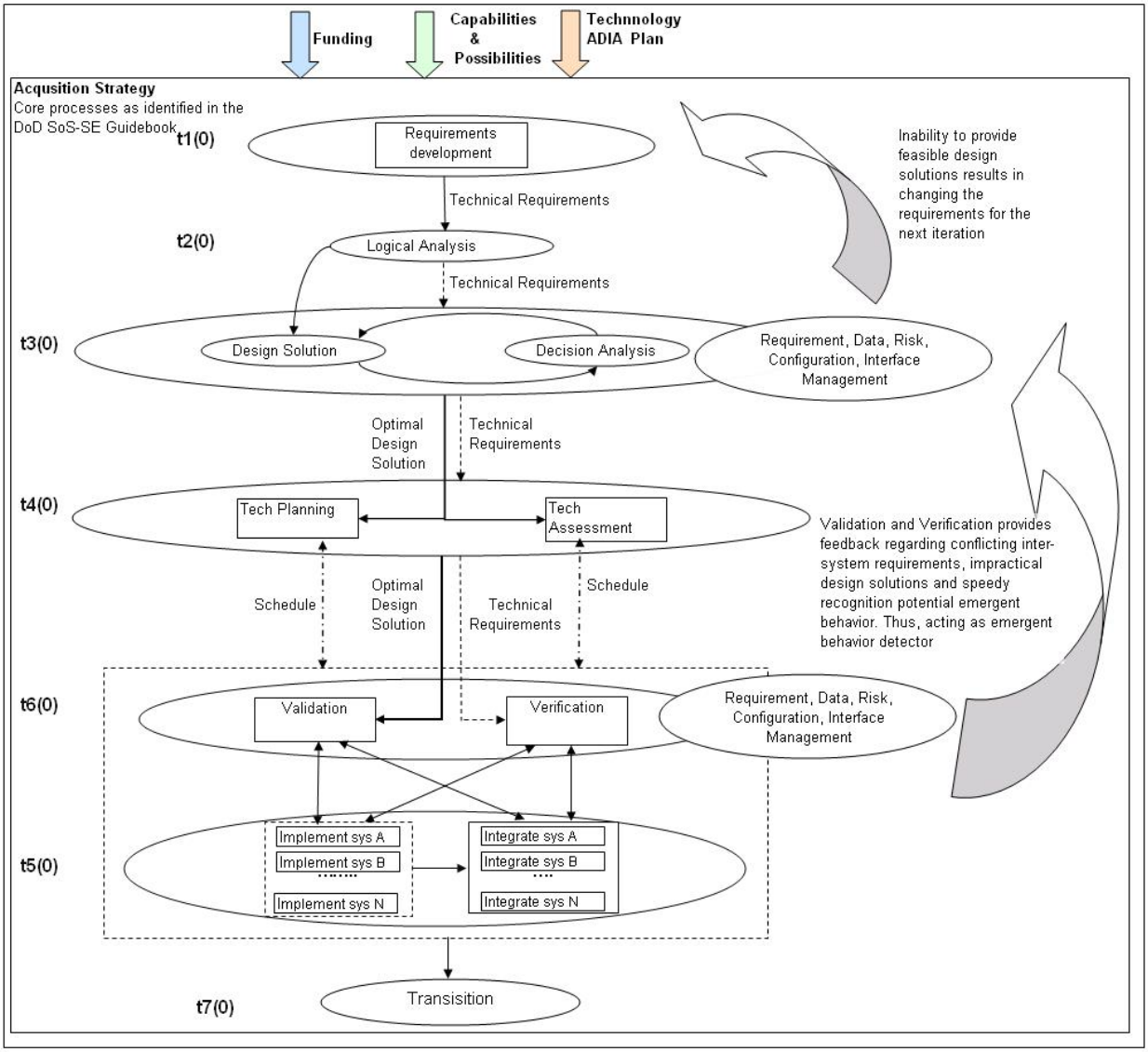

Figure 4. Conceptual Model for Acquisition Strategy Based on SoS SE Processes of Table 1

As can be seen in Figure 4, *Requirements Development* provides the technical requirements of the SoS, based on the relevant external inputs. The pre-acquisition model as discussed in Section 3.1 provides details about the external inputs: CAP, Technology ADIA Plan and funding. The technical requirements are then sent to *Logical Analysis* to check for relationships among the requirements. This also helps to check for inconsistencies among requirements for different systems and how that might affect the functioning and behavior of the future SoS.

Design Solution development and *Decision Analysis* are the next processes that come up with the optimal design solution from the set of feasible solutions to meet the given requirements. The optimal design solution is not only based on the current set of requirements and solution alternatives but it also takes into account all previous information and data available through requirements, risk, configuration, interface and data management processes.

Since most SoS acquisitions are multi-year projects involving many different parties, the overlap between the management processes, *Design Solution* and *Decision Analysis* processes, allows for greater traceability for decisions made. The optimal design solution obtained from this phase is then sent to the next stage: *Technology Planning* and *Technology Assessment*. In the event that there is not an optimal or sub-optimal design solution to successfully implement the given requirements, the feedback loop to *Requirement Development* translates into a change in the technical requirements for the SoS.

Technology Planning and *Technology Assessment* are essentially scheduling processes that help oversee the implementation, integration, validation and verification for all the α-level systems in the SoS. Systems in the SoS are often dependent on other systems for either implementation or integration, or both. These dependencies correspond to time-lags in the acquisition process. For example, if system A is a legacy system and system B is being built, the integration of A with B will not occur until B has been completely implemented. This generates a time lag, especially if another system C is waiting to be implemented based on the integration of A with B. As more systems are added to the SoS, it becomes necessary to generate a schedule that can help coordinate the process. This schedule also needs to be continually updated to reflect unexpected delays, clearly identify bottle-necks, etc.

Due to the heterogeneity of the systems that comprise the SoS and the interactions between then, *Validation* and *Verification* processes need to not only check for suitable implementation of the "optimal design solution" on a system-level but also need to be on the lookout for any misaligned objectives between systems, hidden dependencies among the systems, and any emergent behavior that may affect the functioning and/or behavior of the future SoS. In most situations, early detection of an emergent behavior will prevent the re-designing of major system components and ensure that the SoS functions satisfactorily. Even though *Validation* and *Verification* processes oversee *Implementation* and *Integration*, they occur after *Implementation* and *Integration* have begun.

While *Implementation* and *Integration* are the lowest levels of the acquisition model shown, much of the feedback from this level translates into developing different design solutions and sometimes changing the technical requirements. This level deals with acquiring the systems in the SoS and integrating them based on their dependencies with other systems. These processes consume the bulk of the financial and other resources as well as consume the most time. Therefore, it is understandable why system engineers are often reluctant to re-design functional systems on a whim and why they want to make sure that once the system has been developed, integrated and tested, it does not go back into the *Implementation* phase.

4. Developing a Computational Exploratory Model for Acquisition Strategy

4.1 Overview

Our purpose in constructing a computational exploratory model is to allow acquisition professionals to develop intuition for procuring and deploying system-of-systems. Thus, the objective is not to provide a model validated and ready for deployment in real acquisition programs but to expose the complexities in SoS acquisition. The specific complexities targeted are related to evolutionary development of the SoS and the span-of-control possessed by the SoS managers and engineers. The conceptual model displayed in Figure 4 is implemented using the USCG Integrated Deepwater System as a case study. Given the possibility of

emergent behavior during evolutionary development and the heterogeneity of the components and their interactions, we decided to use the agent based modeling (ABM) approach.

Several challenges arise in transforming the acquisition model to a computational model for purposes of simulation and learning. One challenge lies in converting all the qualitative concepts into quantitative measures to support the computational model for SoS acquisition. We started by building an ideal model devoid of disruptors that historically plague the acquisition process. We will add non-linear behavior to the ideal model to test different scenarios in the process.

A second challenge is building a model that can accommodate the dynamic addition and removal of components in the SoS. In addition, these component systems need to reflect the heterogeneity of the systems in a real acquisition process. We included parameters such as *level of completeness* to demonstrate the difference between legacy systems, new systems and the partially implemented/ integrated systems. *Level of completeness* for both integration and implementation processes vary between 0 and 1, where 1 means that the system is 100% complete. For example, system A representing a fully implemented legacy system has an implementation completeness set to "1," while its integration with system B might only be 20% complete. In this case, the completeness of the integration phase is initialized to 0.20.

A third challenge arises from the numerous methodologies that can be applied to reflect the integration and implementation processes. In a simplified model, it is much easier to begin integration once all the systems have been implemented. However, this method is neither cost nor time efficient, especially in multi-year projects involving numerous systems. On the other hand, dynamically implementing and integrating systems is time efficient but often not possible when dependent systems are outside the span of control of the systems engineers. For example, a cutter and helicopter may be dependent systems being developed by different contractors or different groups that cannot be forced to collaborate. This gives rise to questions: How do we group the systems for integration to achieve maximum efficiency with regard to time and cost incurred? Would it be beneficial to group systems based on the span of control or influence of the systems engineers or on the direct dependencies of the system?

However, developing an acquisition model that studies the effects of all the factors that add to the complexity of the acquisition process for SoS in a short span of time is impossible. Our coarse-scale engineering model will specifically target challenges related to the evolution of the SoS and the span-of-control of the SoS engineer(s).

4.2 Model Inputs

The exploratory computational model developed has a top-down flow with feedback at two junctures. The flow of control begins from Requirement Development (Level t0(0), Figure 4) to Design Solution (Level t3(0), Figure 4) through Logical Analysis(Level t2(0), Figure 4). This linkage is shown in the Figure 5.

The primary inputs fed into the Requirement Development and Logical Analysis processes (e.g., number of requirements, the relationships between these requirements, the systems affected and the dependencies between these systems) are user-defined. The inputs are used to develop a schedule of when each requirement will be implemented, depending on the relationships between the requirements. An example of such user-defined data for these steps is shown in Table 2.

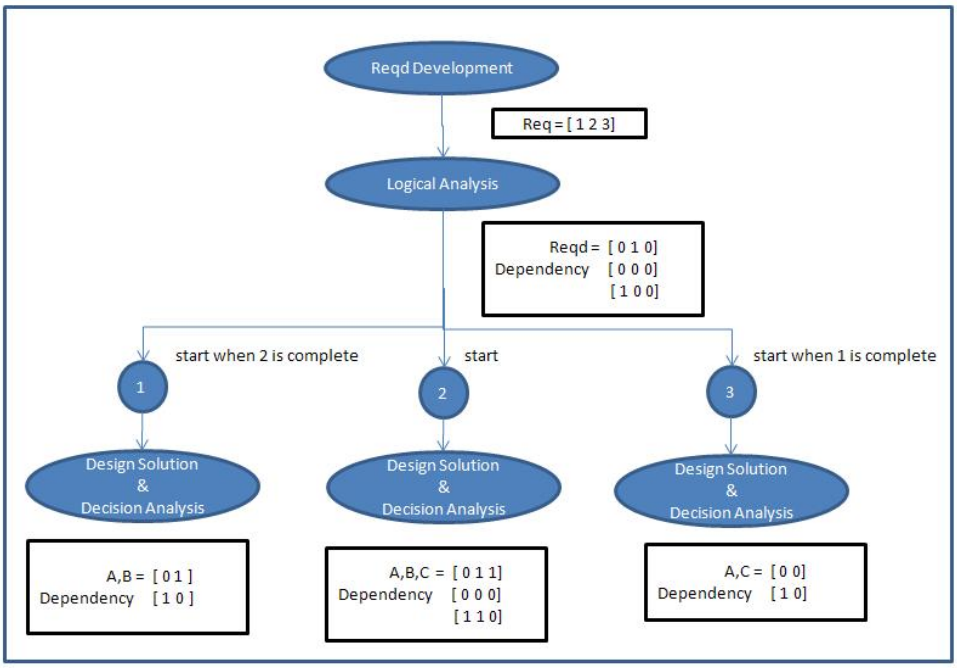

Figure 5. Flow of Control and Parallel Processing of Requirements

As shown in Table 2, there are 3 requirements (1, 2 and 3) and each has a dependency vector associated with it. The vectors are concatenated to form the dependency matrix for requirements. An "X" is placed for all diagonal elements of the dependency matrix because a requirement cannot be dependent on itself. The vector for requirement 1 ([X 1 0]), shows that requirement "1" is dependent on requirement "2." This means that requirement 1 cannot be realized until requirement 2 is implemented. A lack of dependency between requirements means that the requirements can be simultaneously realized. In real world applications, communication upgrade to the North-Atlantic fleet may be independent of the weaponry upgrade for the same group of systems. In such a case, both the requirements on the same group of systems may be implemented simultaneously. Each requirement affects a subset of the systems present in the SoS, and the systems in each subset share a unique dependency matrix with other systems in that subset (shown in Table 2).

Table 2. User-defined Data Used for the Computational Model

Requirements	Dependency	Systems Affected	System-level Dependency
1	[X 1 0]	[A, B]	$\begin{bmatrix} X & 1 \\ 1 & X \end{bmatrix}$
2	[0 X 0]	[A, B, C]	$\begin{bmatrix} X & 1 & 1 \\ 0 & X & 0 \\ 1 & 1 & X \end{bmatrix}$
3	[1 0 X]	[A,C]	$\begin{bmatrix} X & 0 \\ 1 & X \end{bmatrix}$

All component systems of the SoS have user-defined and calculated parameters that expose their heterogeneity and help track their progress through the acquisition process. Some of the parameters used to describe each system in the SoS are described in Table 3. While the

parameters “Name,” “Imp.completeness,” “Imp.time,” “Imp.dependencies,” “Int.completeness,” “Int.time” and “Int.dependencies” are user-defined, ID and Mode are calculated by the model.

Level of completeness for both integration and implementation processes vary between 0 and 1, where 1 means that the system is a 100% complete. A partially complete system may start with a fractional level of completeness. Though the *level of completeness* for implementation and integration are mutually exclusive, a system can not have “0” implementation completeness and a non-zero integration completeness. This qualitatively means that a system that has not yet been implemented cannot be partially integrated with another system. Times to complete implementation and integration are discrete.

Table 3. Parameters Used to Describe Any System in the SoS

Parameter	Description
Name	Name of the system
ID	Unique ID assigned to the system
Imp.completeness[]	An array that gives the completeness of the implementation of the system at any given time.
Imp.dependencies[]	Dependency vector that shows if system implementation is dependant on any other system
Imp.time	Maximum time needed to complete implementation
Int.completeness[]	An array that gives the completeness of the integration of the system with respect to another system at any given time.
Int.dependencies[]	Dependency vector that shows if system integration is dependant on any other system
Int.time	Maximum Time needed to complete integration of system x with any other system
Mode[x;y]	Provides the phase of development the system is currently in and its status

Each system has a pre-defined dependency vector associated with implementation and integration processes. These vectors are concatenated to form a dependency matrix for the systems affected by each requirement. Elements along the diagonal of the dependency matrix (denoted with an “X”) are assigned a value of “0” since a system cannot be dependent or independent of itself. Otherwise, element (i,j) in the system-level dependency matrix can be 0 or 1. A value of 0 signifies that the system (i) is independent of system(j) and a value of 1 signifies that system(i) is dependent on system(j). The dependency matrix can be directed. This occurs when system (i) is dependent on system (j) but not vice-versa. In Table 3, the system-level dependency matrix for requirement 3 is directed.

As previously mentioned, parameters of ID and Mode are assigned by the model. When the system is added to the SoS, it is assigned an ID to uniquely identify it throughout the lifecycle of the SoS. The mode of each system contains two elements: the phase (first element) and the status (second element). Depending on the level of completeness of the system in the integration and implementation phases, the phase of the mode is set to values of {0,1,2,3}. When the system is added into the SoS, 0 is the Mode assigned; 1 and 2 refer to implementation and integration states respectively, and 3 refers to in-operation state. Status of the Mode can be set to 0 or 1, depending on if the system is available for

implementation/integration or not. Thus, a Mode of [2; 1] means that the system is in the integration phase and is currently being integrated with another system.

4.3 Model Dynamics

As seen in Figure 5, the flow of control in the model starts from the *Requirement Development* (Level t0(0), Figure 4) to *Validation and Verification* (Levelt6(0), Figure 4). When the model is first deployed, this stage initializes all processes by supplying requirements to be implemented, systems affected, etc. Each requirement also has a "change" status flag that shows when a particular requirement is unchanged (status=0), being changed (status=-1) or has been changed (status=1). When a requirement is changed after the acquisition process has begun, it affects all subsequent processes.

Using the user-defined parameters and inputs from *Requirement Development* (similar to the data shown in Table 2), *Logical Analysis* (Level t2(0), Figure 4*)* generates a schedule to realize the given requirements. Depending on the dependencies between the requirements, they get implemented in series or in parallel with other requirements. As shown in Figure 5, every requirement being implemented is fed into its own *Design Solution* and *Decision Analysis* (Level t3(0), Figure 4*)* process. This process can change the implementation or integration times required by each system affected by the requirement. If a requirement is changed, the design solution and implementation processes for the component systems are affected. The set of systems with their modified implementation and integration times are then sent through the *Technology Planning* and *Technology Assessment* (Level t4(0), Figure 4) processes.

Technology Planning takes in the array of systems being affected by the given requirement and divides them into "to-be-implemented" and "read-for-integration" queues depending on the implementation/integration times needed for each system. Since the component systems in the acquisition process can be at varying levels of completeness during the implementation and integration cycles, they are dynamically added to the queues. For each queue, a synchronization matrix (sync_mat) is generated to keep track of the number of systems in the queue, their expected times of completion and their "iteration-rate." Given the maximum time allotted and the existing level of completeness, Iteration rate is defined as the average rate at which the process needs to be completed for a system. For example, if system "A," which is 25% completed needs to be fully implemented in 5 time-steps, using Equation 1, the iteration rate of system "A" is calculated to be 0.15.

Equation 1. $$Iteration\_Rate = \frac{1 - completeness(0)}{\max\_time}$$

The systems in the synchronization matrix for the implementation/integration queues are sorted on the basis of 1) number of systems dependent on that system and, 2) maximum time required by the system to complete the process. Thus, systems with larger number of systems dependent on them are higher on the synchronization queue than those with a fewer number of dependent systems. Similarly, systems with the same number of dependent systems—those requiring less time for implementation/integration—get higher priority. Since this is a basic model, we only have two criteria to determine an appropriate schedule. In future models, more conditions can be added to simulate different methodologies. For example, priority of the systems in the queue may be a factor that determines their position in the process queues.

Technology Assessment uses the synchronization matrix developed by *Technology Planning* to track the progress of the systems in the implementation/ integration queues. In a

non-ideal acquisition model, the iteration rates for the systems will be subject to external perturbations. A drastically reduced iteration-rate will stall the development of a system mid-process and affect other systems dependent on the stalled system. *Technology Assessment* recognizes the stalled systems and activates "enablers" to re-adjust their iteration-rate.

Implementation (Level t5(0), Figure 4) of systems can also occur in series or parallel configurations depending on the system dependencies. The level of completeness for an implementation process increases by the iteration rate at every time-step, until it reaches a completeness value of 1. The incremental increase in the level of completeness of two independent systems with different iteration rates occurs simultaneously, as shown in Figure 6a. If system "B" were dependent on system "A," then implementation of B would commence when A was fully implemented, as shown in Figure 6b. When a system achieves the implementation completeness value of 1, it is added onto the integration queue. Since the integration and implementation process queues are dynamically generated, the synchronization matrix for the systems in the queues also changes dynamically.

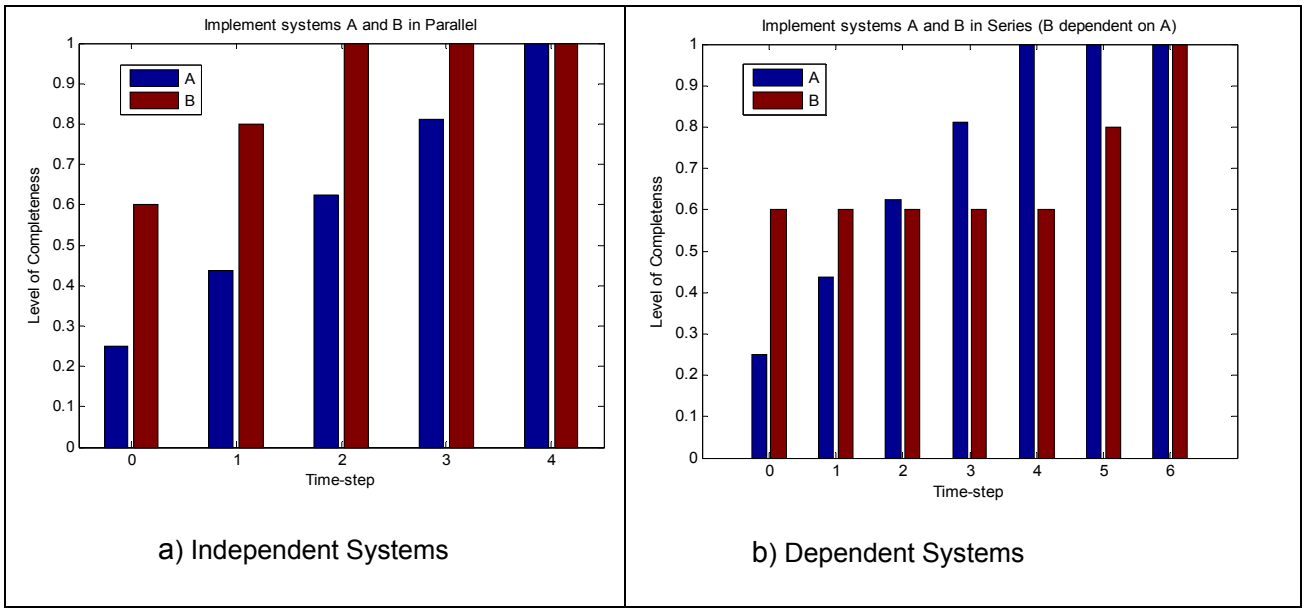

Figure 6. Incremental Increase in Implementation Completeness for a) Independent Systems b) Dependent Systems.

The *Integration* (Level t5(0), Figure 4) process also uses a synchronization matrix for coordination. While parallel processing of dynamically changing process queues greatly improves the time efficiency for independent systems, they also increase the probability of dependent systems trying to integrate with each other at the same time. For example, system "A" might be waiting to be integrated with system "B" while system "B" is being integrated with system "C." The current implementation algorithm solves this problem by using the Mode parameter for each system. As described in Table 3, Mode is a 2x1 array structure. The first element gives the phase the system is currently in and the second element gives the status. When a system in the integration queue (phase 2) starts getting integrated with another system its status changes from 0 to 1. In the previous example, when "B" begins integration with system "C," the mode status of "B" changes to 1. System "A" cannot integrate with system "B," unless the mode status for "B" changes back to 0. This process is better illustrated in Figure 7.

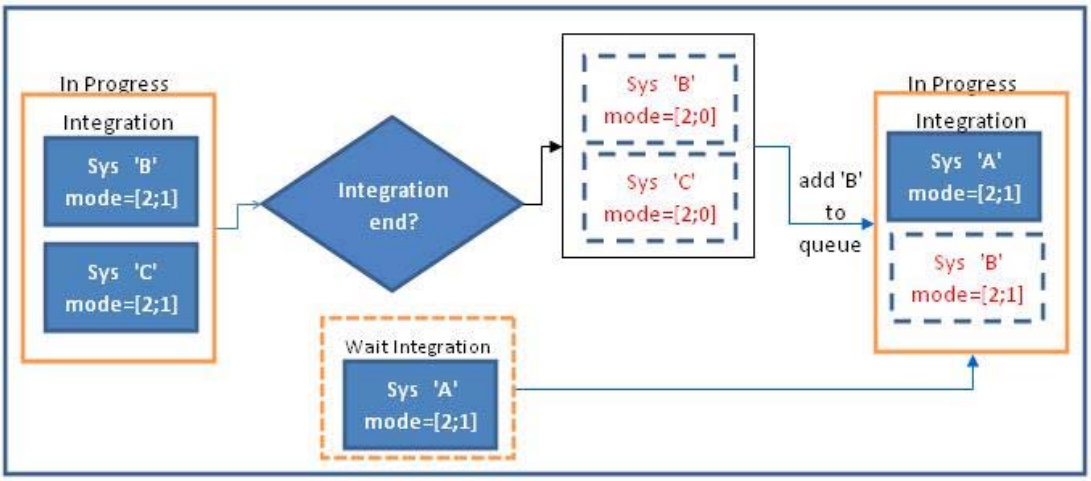

Figure 7. Use of Mode in the Integration Process Queue

When both the *Implementation* and *Integration* processes for all the systems affected are complete, the *Validation* and *Verification* (Level t6(0), Figure 4) processes check to see that all the systems were implemented and integrated within constraints of time and budget. They check for a completeness level of 1 for all the component systems affected, and then they compare the actual time needed and cost incurred by each system to the time and cost allotted to them by *Design Solution* and *Decision Analysis*.

5. Work in Progress: Implementation and Success Metrics

As discussed in Section 4, the modeling approach for acquisition allows for a great deal of flexibility in terms of parameters, methodologies, etc. While we have generated a successful "small-scale" ideal model to depict the basic methodology of the defense acquisition process, we have yet to implement different scenarios using disruptors and enablers.

Also, while the feedback system within the system-level has been implemented between processes like integration and technology planning, the feedback system between *Design Solution* and *Requirements Development* has yet to be implemented. The high-level feedback loops to *Requirements Development* and *Logical Analysis* will allow us to model scenarios in which requirements change at different times through out the lifecycle of the SoS. As mentioned in Section 2, historical trends in defense and non-defense related acquisition processes show that requirement changes occurring during the later phases of the acquisition lifecycle have been major contributors to the failure of the SoS acquisition program.

We are also currently working on developing the *soft* parameters that allow for fuzzy boundaries depicting varying spans of control of the SoS engineers and managers over the different component systems This may be reflected in the system-dependency matrix by using values between 0 and 1. Fractional values of dependency then need to be mapped to a time-delay parameter in each process, in order to show the relationship (if any) between the span of authority of a systems or system-of-systems engineer and the time required for the completion of an acquisition process.

The success of the developed ABM Model will be measured by the ability of the model to provide insights into the types of acquisition management insights and approaches that can increase the success of an acquisition process in the SoS setting. The model will enable us to answer the three research questions posed in Section 1, specifically targeting complexities related to span of control/influence of the SoS engineers and managers and the evolutionary development of an SoS. A successful model will allow for the study of various scenarios generated by implementing different acquisition management strategies and approaches in an SoS environment. For example, scenarios could be generated by adjusting the "levels" of collaboration between individual system engineers at varying hierarchical levels or the span of control of the SoS engineer throughout the lifecycle of the SoS. The model may even be used as a comparative tool to study the effect of implementing different methodologies for individual processes on SoS parameters, such as time needed to acquire, test and deploy specific capabilities. The greatest benefit of such modeling lies in its ability to act as a decision-making tool for SoS engineers, program managers and systems engineers to improve the probability of success of the SoS acquisition process by simulating different scenarios and implementing different strategies.

6. Conclusions

From historical data related to past SoS-oriented defense acquisition programs (discussed in Section 2), we summarize the common causes of failure as a) misalignment of objectives among the systems, b) limited span of control of the SoS engineer on the component systems of the SoS, c) evolution of the SoS, d) inflexibility of the component system designs, e) emergent behavior revealing hidden dependencies within systems, f) perceived complexity of systems, and g) the challenges in accurately representing them. These sources of complexity were mapped to a section of the SoSE Process Model recently introduced by Sage and Biemer (2007) in order to identify where manifestations of these complexities might arise and how to begin assessment of how they may impact the acquisition process.

This mapping in conjunction with the 16 technical and technical management SE processes identified by the DoD SoS-SE Guide (Systems and Software Engineering, 2006) was used to develop a conceptual model for pre-acquisition and acquisition strategy activities. The acquisition strategy model takes an incremental approach to the evolutionary development of an SoS and allows processes lower in the hierarchy to affect change in the processes above them. Thus, the model exposes the interconnections among levels and uses these to implement evolving requirements and design solutions in the component systems of the SoS.

These mappings and conceptual models are directed toward providing a basis for a computational exploratory model for acquisition strategy in an SoS environment. Based on user-defined inputs for the requirements and their dependencies on each other, the model uses series and parallel processing to implement these requirements in the component systems. This exploratory model allows evolving requirements and design solutions to trickle through the lower processes and uses disruptors to affect specific component systems, which in-turn affects change in processes higher in the hierarchy.

The uniqueness of the models lie in their ability to provide a better understanding of the acquisition process in an SoS environment, along with computational tools for better decision-making for the higher levels of SoS management. We hope the insights gained from this research will significantly improve the probability of success with future acquisition programs within and without the DoD.

List of References

Allen, T. (2007, April). *Statement on the converted 123-foot patrol boats and changes to the deepwater acquisition program*. USCG Press Release. Retrieved June 1, 2007, from www.piersystem.com/go/doc/786/154307

Anderson, M, Burton, D, Palmquist, M.S., & Watson, J.M. (1999, May). The deepwater project—A sea of change for the US coast guard. *Naval Engineers Journal*. Retrieved April 1, 2008, from http://www.prosoft.tv/aseaofchange.pdf

Bar-Yam, Y. (2003, April). *Complexity of military conflict: Multiscale complex systems analysis of littoral warfare*. New England Complex Systems Institute. Retrieved March 2008, from http://www.necsi.edu/projects/yaneer/SSG_NECSI_3_Litt.pdf

Caldwell, S.L. (2006, June).*Coast guard: Observations on agency performance, operations and future challenges*. Washington, DC: GAO. Retrieved January 3, 2008, from http://www.gao.gov/new.items/d06448t.pdf

Capaccio, T. (2006, December). *Boeing systems delayed, 11 others killed in proposed army budget*. Retrieved June 1, 2007, from http:// www.bloomberg.com/apps/news?pid=newsarchive&sid=amNoYrTtynxQ

Committee on Systems Integration for Project Constellation. (2004 ,September). *Systems integration for project constellation*. The National Academies. Retrieved March 2008, from http://nap.edu/html/proj-constellation/ltr-rep.pdf

DoD. (2006). *Defense acquisition guidebook*. Defense Acquisition Guidebook website. Retrieved April 4, 2008, from https://akss.dau.mil/dag/TOC_GuideBook.asp?sNode=R2-3-2&Exp=Y.

Fowler, C.A. (1994, August). The defense acquisition system too late for the scalpel; bring out the meataxe! *IEEE Aerospace and Electronic Systems Magazine*, *9*(8), 3-6.

Maier, M. (1998). Architecting principles for system-of-systems. *Systems Engineering*, *1*, 267-284.

Polzer, H., DeLaurentis, D.A., & Fry, D.N. (2007, April). Multiplicity of perspectives, context scope, and context shifting events. In *Proceedings of the 2007 IEEE International Conference on System of Systems Engineering* (pp. 1-6).

Riccioni, E. (2005, March). *Description of our failing defense acquisition system as exemplified by the history, nature and analysis of the USAF F-22 raptor program*. Project on Government Oversight. Retrieved March 4, 2008, from http://www.pogo.org/m/dp/dp-fa22-Riccioni-03082005.pdf

Rouse, W. (2007, June). Complex engineered, organizational and natural systems. *Systems Engineering*, *10*(3), 260-271.

Sage, A., & Biemer, S. (2007). Processes for system family architecting, design, and integration. *IEEE Systems Journal*, *1*(1), 5-16.

Saunders, T., Croom, C., Austin, W., Brock, J., Crawford, N., Endsley, M., et al. (2005, July). *System-of-systems engineering for air force capability development* (SAB-TR-05-04). Washington, DC: USAF Scientific Advisory Board.

Simon, H. (1996). *Sciences of the artificial*. Cambridge, MA: MIT Press.

Spring, B. (2005, October). *Congressional restraint is key to successful defense acquisition reform*. The Heritage Foundation. Retrieved March 26, 2008, from http://www.heritage.org/Research/NationalSecurity/bg1885.cfm

Systems and Software Engineering. (2006). *SoS systems engineering*. Retrieved March 2, 2008, from http://www.acq.osd.mil/sse/ssa/initiat_sos-se.html

United States Coast Guard Acquisition Directorate. (1998). *Coast guard recapitalization fact sheet*. Retrieved January 3, 2008, from http://www.uscg.mil/acquisition/programs/pdf/CG-9recap.pdf

Acknowledgements

The authors are grateful for the sponsorship of this work by the Naval Postgraduate School's Acquisition Research Program. The authors also wish to acknowledge and express their gratitude to Chris Patterson and Yury Pensky for their contribution to the project.

2003 - 2008 Sponsored Research Topics

Acquisition Management

- Software Requirements for OA
- Managing Services Supply Chain
- Acquiring Combat Capability via Public-Private Partnerships (PPPs)
- Knowledge Value Added (KVA) + Real Options (RO) Applied to Shipyard Planning Processes
- Portfolio Optimization via KVA + RO
- MOSA Contracting Implications
- Strategy for Defense Acquisition Research
- Spiral Development
- BCA: Contractor vs. Organic Growth

Contract Management

- USAF IT Commodity Council
- Contractors in 21st Century Combat Zone
- Joint Contingency Contracting
- Navy Contract Writing Guide
- Commodity Sourcing Strategies
- Past Performance in Source Selection
- USMC Contingency Contracting
- Transforming DoD Contract Closeout
- Model for Optimizing Contingency Contracting Planning and Execution

Financial Management

- PPPs and Government Financing
- Energy Saving Contracts/DoD Mobile Assets
- Capital Budgeting for DoD
- Financing DoD Budget via PPPs
- ROI of Information Warfare Systems
- Acquisitions via leasing: MPS case
- Special Termination Liability in MDAPs

Human Resources

- Learning Management Systems
- Tuition Assistance
- Retention
- Indefinite Reenlistment
- Individual Augmentation

Logistics Management

- R-TOC Aegis Microwave Power Tubes
- Privatization-NOSL/NAWCI
- Army LOG MOD
- PBL (4)
- Contractors Supporting Military Operations
- RFID (4)
- Strategic Sourcing
- ASDS Product Support Analysis
- Analysis of LAV Depot Maintenance
- Diffusion/Variability on Vendor Performance Evaluation
- Optimizing CIWS Lifecycle Support (LCS)

Program Management

- Building Collaborative Capacity
- Knowledge, Responsibilities and Decision Rights in MDAPs
- KVA Applied to Aegis and SSDS
- Business Process Reengineering (BPR) for LCS Mission Module Acquisition
- Terminating Your Own Program
- Collaborative IT Tools Leveraging Competence

A complete listing and electronic copies of published research are available on our website: www.acquisitionresearch.org

ACQUISITION RESEARCH PROGRAM
GRADUATE SCHOOL OF BUSINESS & PUBLIC POLICY
NAVAL POSTGRADUATE SCHOOL
555 DYER ROAD, INGERSOLL HALL
MONTEREY, CALIFORNIA 93943

www.acquisitionresearch.org